НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2019р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 “ Програмна інженерія “

на тему: Освітня web-система моделювання процесів створення розумного будинку з елементами віртуальної реальності

Виконав: студент 4 курсу, групи__ТВ-з51

_____Василенко Максим Олегович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

Напрям підготовки 6.050103 “ Програмна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019р.

ЗАВДАННЯ
на дипломну роботу студенту
Василенко Максима Олеговича
(прізвище, ім’я, по батькові)

1.Тема роботи Освітня web-система моделювання процесів створення розумного будинку з елементами віртуальної реальності
Керівник роботи Шульженко Олег Феодосійович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ” ” 201 р. № __

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи: мова програмування Javascript.
4.Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) Розробити освітню web-систему моделювання процесів створення розумного будинку з елементами віртуальної реальності. Розробити програмний інтерфейс платформи.

5. Перелік ілюстративного матеріалу: схеми архітектури додатку, діаграма прецедентів системи, діаграма структури системи, зразки розробленого інтерфейсу додатку.
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ” ”_____201 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі		
2	Розробка архітектури та загальної структури системи		
3.	Розробка структур окремих підсистем		
4.	Програмна реалізація системи		
5.	Оформлення пояснювальної записки		
6.	1 Захист програмного продукту		
7.	2 Передзахист		
8.	Захист		

Студент _____ Василенко М. О.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Шульженко О. Ф.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Подана дипломна робота присвячена розробці освітньої web-системи моделювання процесів створення розумного будинку з елементами віртуальної реальності.

Метою роботи є створення освітньої web – системи моделювання процесів створення розумного будинку з елементами віртуальної реальності.

Для досягнення мети були вирішені наступні задачі:

1. Спроектовано архітектуру освітньої платформи.
2. Створено web-систему моделювання процесів створення розумного будинку з елементами віртуальної реальності.
3. Інтегровано мобільний додаток, який дозволяє взаємодіяти з VR – сценою.
4. Побудовано інтерфейс, що дозволяє користувачу проходити освітні курси.
5. Побудовано персональний кабінет користувача, в якому можна відслідкувати прогрес по курсам.
6. Курси класифіковано по рівню складності.
7. Побудовано інтерфейс для створення власних курсів.

Ключові слова: освітня платформа, web-система, розумний будинок, віртуальна реальність.

Обсяг звіту становить 77 сторінок, міститься 15 ілюстрацій, 3 додатки. Загалом опрацьовано 34 джерел.

ABSTRACT

The thesis is devoted to the development of an educational web-system for modeling the processes of creating a smart home with elements of virtual reality.

The purpose of the work is to create an educational web - system for modeling the processes of creating a smart home with elements of virtual reality.

To achieve the goal, the following tasks were solved:

1. An interface is created that allows the user to pass educational courses.

2. A personal user room has been built in which you can track the progress of the courses.

3. Courses are classified according to the level of difficulty.

4. Integrated mobile application that allows you to interact with the VR scene.

5. An interface was created for creating your own courses.

Key words: educational system, web-platform, intelligent house, virtual reality.

The report volume is 77 pages, contains 15 illustrations, 3 attachments. In total, 34 sources have been processed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

TTD	–	Time Travel Debugging
VR	–	Virtual reality
CoR	–	Chain of Responsibility
Фреймворк	–	Інфраструктура програмних рішень, що полегшує розробку складних систем
VR	–	Virtual reality
RESTful	–	Representational State Transfer
БД	–	база даних
ORM	–	Object-Relational Mapping

ВСТУП

Сучасні технології інтегруються в різні сфери нашого життя щодня. Одна з таких сфер – освіта, що розвивається щодня, розробляються нові методи для покращення рівню освіти. Саме тому інтеграція сучасних технологій в освітній процес сприяє зміні підходу до навчання, збільшенню об’єму поглинутих знань, використанню нові технології для самореалізації у не доступних раніше сферах.

Багато освітніх методів створені завдяки використанню комп’ютерів, смартфонів, планшетів. З появою віддаленого навчання, завдяки освітнім платформам та застосункам, що дозволяють користувачу набувати знань по підготовленим програмам, можливості збільшуються. За рахунок щохвилинної доступності до навчального матеріалу та відсутності фіксованого часу для навчання, користувачі мають можливість навчатись у будь-який час. Це сприяє навчальному процесу та дозволяє розвиватись освітнім методам у новому руслі.

Нові формати представлення інформації дозволяють експериментувати з створенням персональних навчальних підходів, що сприяє підвищенню продуктивності навчання та збільшує кількість засвоєного матеріалу. Одним з таких форматів є освітні web-платформи, що надають можливість охопити всі вікові категорії, персоналізуючи освітній матеріал під вік користувача та подаючи навчання в різних формах: лекцій, відео-матеріалів, ігрових застосунків.

Також в наш час набуває популярності віртуальна реальність та інтеграція її у різні сфери життя: навчання, відпочинок, розваги. Перспективи використання віртуальної реальності в промисловості, фінансовій сфері, навчанні прогнозують серйозне полегшення існуючих процесів. Інтеграція віртуальної реальності в освітній процес у тому числі відкриває нові освітні методи та навчальні підходи.

Метою дипломної роботи є реалізація комбінації описаних вище підходів і буде розглянута у поданому дослідженні на базі освітньої web-платформи моделювання процесів створення розумного будинку з елементами віртуальної реальності, яка

сприятиме підвищенню рівня навчання окремої сфери, допоможе проводити практичні зайняття без ризику для здоров’я та особливих витрат.

Головними задачами дослідження є:

1. Вивчення й аналіз існуючих систем, які дозволяють інтегрувати в навчальний процес елементи віртуальної реальності.

2. Вивчення й аналіз новітніх підходів до інтегрування в освітній процес елементів віртуальної реальності.

3. Порівняльний аналіз конкурентних систем та практик.

4. Розробка власного варіанту системи інтегрування елементів віртуальної реальності на базі найкращих практик, підходів та інструментів.

Впровадження запропонованої системи інтеграції елементів віртуальної реальності відкриває можливість впровадження віртуальної реальності у освітній процес та використовувати її як один із способів підвищення якості навчального процесу.

Тому, було запропоновано створити освітню Web-систему з елементами віртуальної реальності для вивчення та дослідження процесів розумного дому.

1. СТВОРЕННЯ ОСВІТНЬОЇ WEB-СИСТЕМИ МОДЕЛЮВАННЯ ПРОЦЕСІВ СТВОРЕННЯ РОЗУМНОГО БУДИНКУ З ЕЛЕМЕНТАМИ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ

В останні роки в нашому світі гостро постало питання якісної освіти. У сучасному світі в останні десятиріччя широкого поширення набули проблеми якісного навчання, розробка нових освітніх методів, адже інформацію, яку потрібно вивчити людині, вона повинна вивчити швидко, але дуже добре запам’ятати і подалі мати можливість користуватися нею. Тому у навчальний процес прийшли новітні технології, які дають змогу більш глибоко зануритися у освітній процес, більш детально вивчити матеріал.

Дуже важливим пунктом в освітньому процесі є практичні зайняття – вони дають змогу не тільки закріпити знання, а й показати поведінку у реальному житті.

Важливою проблемою, яку вирішують за допомогою віртуальної реальності в освіті, є надання можливості для практичних зайнять, недоступні у реальному житті через небезпеку або високу вартість. Віртуальна реальність дає змогу пробувати, тестувати набуті знання у віртуальному світі, який майже нічим не відрізняється від реального.

Метою даної дипломної роботи є створення освітньої web – системи моделювання процесів створення розумного будинку з елементами віртуальної реальності.

При розробці відповідного програмного забезпечення постали наступні завдання:

1. Побудувати інтерфейс що дозволяє проходити освітні курси.
2. Побудувати інтерфейс персонального кабінету користувача.
3. Класифікувати курси по рівню складності.

4. Побудувати інтерфейс для створення нових курсів.

5. Інтегрувати мобільний додаток, який дозволяє взаємодіяти з VR – сценою.

Тому було запропоновано дослідити методи інтеграції віртуальної реальності та можливість їх застосування для вирішення поставленої задачі.

Як результат дослідження, потрібно створити програмний застосунок, основними функціями якого є:

- дати можливість створювати курси для моделювання процесів розумного будинку;
- створення анімацій та завдань для виконання у віртуальній реальності;
- дати можливість користувачам проходити курси;
- дати можливість дивитися анімації у віртуальній реальності.
- надати можливість проходити завдання у віртуальній реальності.
- забезпечити користувача системи графічним інтерфейсом, який може працювати як на мобільному телефоні, так і на комп’ютері.

1.1 Огляд існуючих освітніх web-платформ

З кожним роком набувають все більшого поширення різноманітні освітні системи, а особливо неінституціональна модель освіти, що орієнтується на організацію. освіти поза школами та ВНЗ, зокрема за допомогою Інтернету. Їх створюють та підтримують великі організації, університети, вчені, люди, які хочуть розвивати освітні методи. Одною з тем, які можна вивчити за допомогою цих систем є моделювання процесів розумного будинку.

З метою кращого розуміння теми розумних будинків, а також більш глибокого занурення в цю тему за допомогою залучення практичних зайнять було запропоновано розробити освітню Web-систему із елементами віртуальної реальності. Це допоможе більш глибоко вивчити процеси, які проходять у розумному

будинку і «доторкнутися» до сенсорів та того, як працюють розумні будинки, не купляючи дороге обладнання.

1.2 Аналіз існуючих освітніх web-платформ

Під час пошуку інформації, та аналізу існуючих рішень, було зроблено висновок, що на даний момент такі системи мають свої недоліки, а також не дозволяють спробувати набуті знання у віртуальній реальності:

— “Udemy” — це освітня Web-система, яка спеціалізується на курсах по різноманітних темах, її особливістю, а також недоліком є те, що майже кожен може розробити свій курс, не зважаючи на його корисність, а також те, що ці проходження курсів на цій системі не має ніякого юридичного підтвердження;

— “Coursera” — це платформа, на якій університети можуть створювати свої курси. На ній після проходження курсу видається сертифікат, але зв’язок із розробниками курсу досить обмежений. Також недоліком є те, що курси з теми розумного будинку не мають ніяких практичних уроків.

Також недоліком цих систем є їх громіздкість. На цих платформах дуже багато різноманітних курсів, тому досить складно знайти потрібний освітній курс.

Оскільки IoT, розумні будинки вже повільно інтегруються в наше життя і у недалекому майбутньому усі люди будуть жити в таких будинках, то розробка освітньої системи із можливістю не тільки вивчити цю тему, а й самому спробувати створити свій розумний дім у віртуальній реальності є актуальною задачею.

Розробка програмної системи, заснованої на веб-технологіях, є одним з варіантів, що працюватиме на будь-якому пристрої.

1.3 Висновки до розділу

У розділі “ Створення освітньої web-платформи моделювання процесів створення розумного будинку з елементами віртуальної реальності ” було описано результати аналізу існуючих освітніх платформ та технологій, які використовувались при їх розробці.

2. ЗАСОБИ РЕАЛІЗАЦІЇ ОСВІТНЬОЇ WEB-СИСТЕМИ

Під час аналізу поданої задачі та методів її вирішення, було прийняте рішення створювати програмний комплекс на базі зв’язки веб-технологій та мобільного застосунку. Головною перевагою веб-застосунку є можливість його використання на будь-яких пристроях не залежно від операційної системи. Мобільний застосунок використовується як пристрій для відображення віртуальної реальності, адже у багатьох людей є смартфон, а купити звичайні окуляри, наприклад Cardboard, дешевше, аніж покупати дорогі окуляри як наприклад Oculus Rift або HTC Vive.

2.1 Вибір архітектури програмного комплексу

Для реалізації задачі було прийняте рішення використати наступну архітектуру: веб-застосунок, мобільний застосунок, сервер і база даних. Схема обраної архітектури зображена на рисунку 2.1.

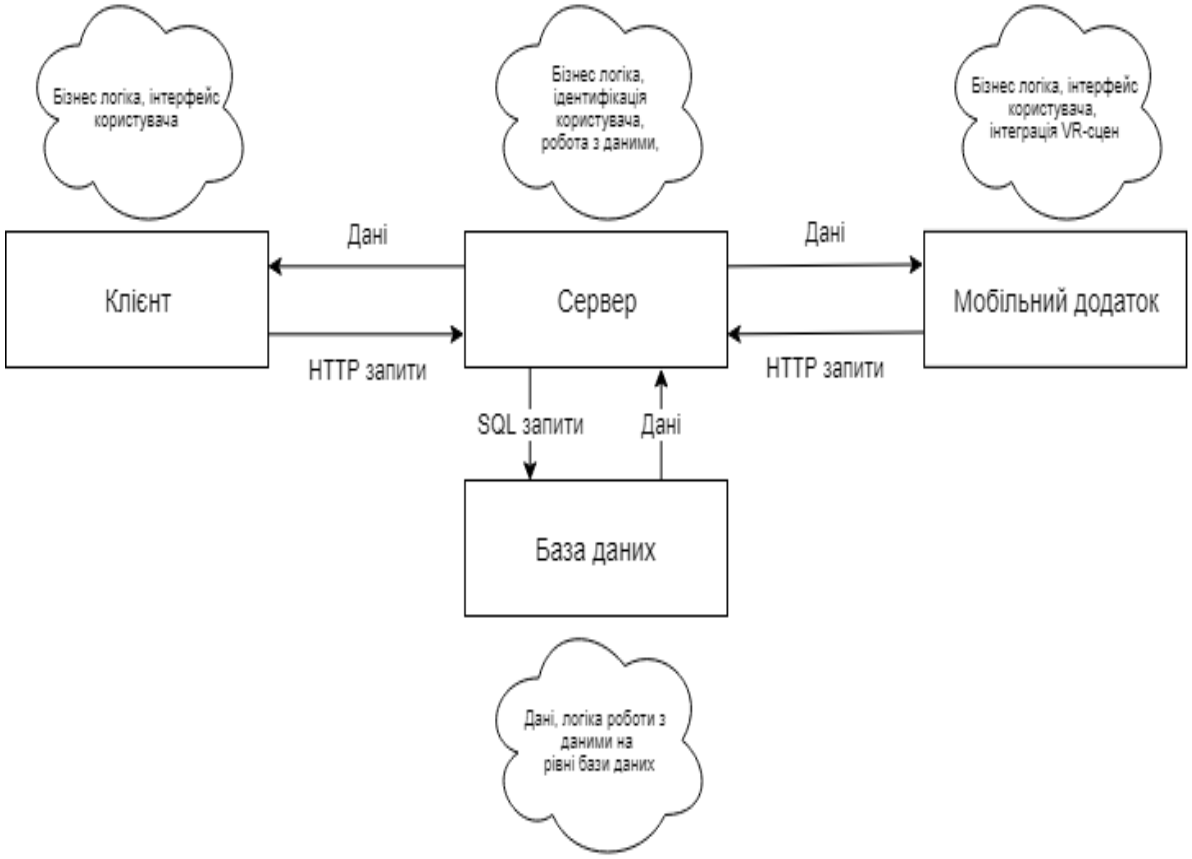

Рисунок 2.1 – Архітектура програмного комплексу

Головним в обраній архітектурі є сервер. Він зосереджує у собі основну бізнес-логіку та контролює доступ до бази даних. Для надання персонального доступу до додатку відбувається ідентифікація користувача за допомогою серверу. Єдиним зв’язком між базою даних та користувачем є сервер, щоб попередити можливе використання даних не за призначенням та їх пошкодження. Для того, щоб користуватися програмою, користувач має бути авторизованим у системі, тому дана логіка реалізується на рівні серверу.

При користуванні освітньою системою, користувач взаємодіє з клієнтським додатком, веб-сайтом в даному випадку. Для користувача реалізовано інтерфейс, за допомогою якого відбувається налаштування програми та перегляд курсів. Також на користувацькому рівні відбувається первинна обробка даних перед відправленням на сервер і опрацювання результатів, отриманих від сервера. Ще на цьому рівні

відбувається перший етап автентифікації користувача для обмеження неконтрольованого доступу до програми.

Під час проходження практичної частини курсів, користувач взаємодіє із мобільним застосунком. У ньому реалізований інтерфейс автентифікації і інтерфейс взаємодії користувача з віртуальною реальністю.

Сервер зберігає дані для подальшого використання на рівні бази даних. Цілісність даних забезпечується за допомогою зовнішніх зв’язків та ключів. Частину бізнес-логіки можна реалізувати на рівні самої бази даних, якщо вона не потребує використання інших джерел.

2.2 Опис архітектури серверної частини

Шаблон проектування — це архітектурна конструкція, що являє собою рішення задач проектування, які часто зустрічаються в рамках певного контексту.

Для серверної частини додатку було обрано фреймворк, який реалізую шаблон проектування CoR. [1]

Шаблон CoR (Chain of Responsibility) — це шаблон проектування, головною ідеєю якого є розділити обробників запитів один від одного, які самі вирішують, чи можуть вони обробити цей запит (рисунок 2.2). Принципом CoR є розділення усіх обробників запитів, а сам запит передається як об’єкт із параметрами.

Цей шаблон є поведінковим, тому він вирішує, як саме програма повинна поводити себе у різних ситуаціях. Він пропонує зв’язати усіх обробників запитів у одну ланку. Кожен обробник буде мати в собі посилання на наступний обробник. Таким чином, кожен обробник може не тільки щось зробити із запитом, але й передати управління наступному обробникові.

Передаючи запити у перший обробник у ланці можна бути впевненим, що запит надійде і до останнього обробника. Причому довжина цієї ланки не має ніякого значення.

Також обробник може не передавати запит далі. Тому можна побудувати таку систему, в якій деякі обробники будуть робити щось із запитом, наприклад переводити параметри запиту у спеціальний формат, передавати ці дані далі, а наступний обробник зробить якійсь дії із базою даних та відправить їх далі, а вже останній обробник відповість клієнту на запит.

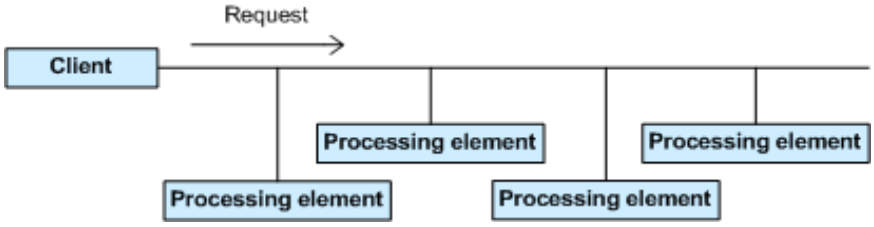

Рисунок 2.2 – Схема роботи CoR шаблону

Основна мета застосування цього шаблону – зручність у розширенні наявних обробників запитів, замість додавання розгалуження та складної системи. У такий спосіб сервер можна поділити на умовні частини, коли кожна частина буде відповідати за окрему частку запитів.[16]

Шаблон Chain of Responsibility дозволяє уникнути залежності відправника запиту від його одержувача, при цьому запит може бути оброблений кількома об'єктами. Об'єкти-одержувачі зв'язуються в ланцюжок. Запит передається по цьому ланцюжку, поки не буде оброблений.[16]

Даний шаблон вводить конвеєрну обробку для запиту з безліччю можливих оброблювачів.

Похідні класи знають, як обробляти запити клієнтів. Якщо "поточний" об'єкт не розуміє запит, то він делегує його базовому класу, який делегує "наступного" об'єкту і так далі.

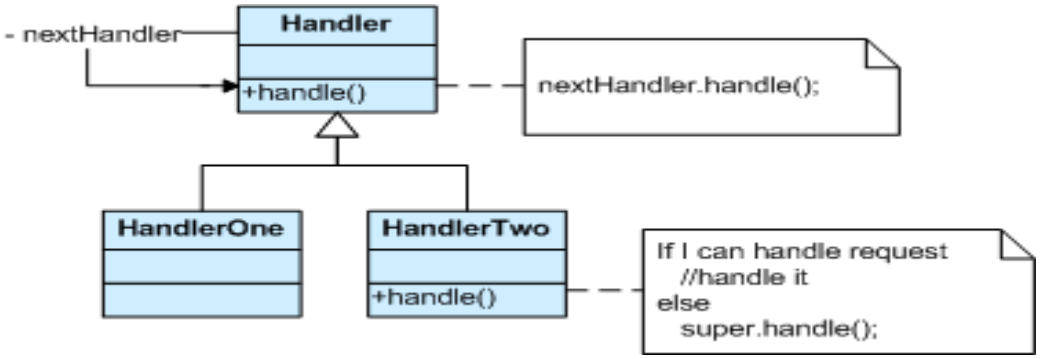

Рисунок 2.3 – UML-діаграма класів CoR шаблону

Обробники можуть вносити свій вклад в обробку кожного запиту. Запит може бути переданий по всій довжині ланцюжка до самого останнього ланки.

2.3 Опис архітектури клієнтського застосунку

Для реалізації клієнтського застосунку було обрано фреймворк, в основі якого шаблон проектування MVC (рисунок 2.4). [1]

MVC (Model-View-Controller) — це шаблон проектування, що передбачає поділ системи на три взаємопов'язані частини: M — Model (модель даних), V — View (інтерфейс користувача) та C — Controller (модуль керування). головною ідеєю якого є відділити логіку застосунку від представлення (рисунок 3.2). Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача. [34]

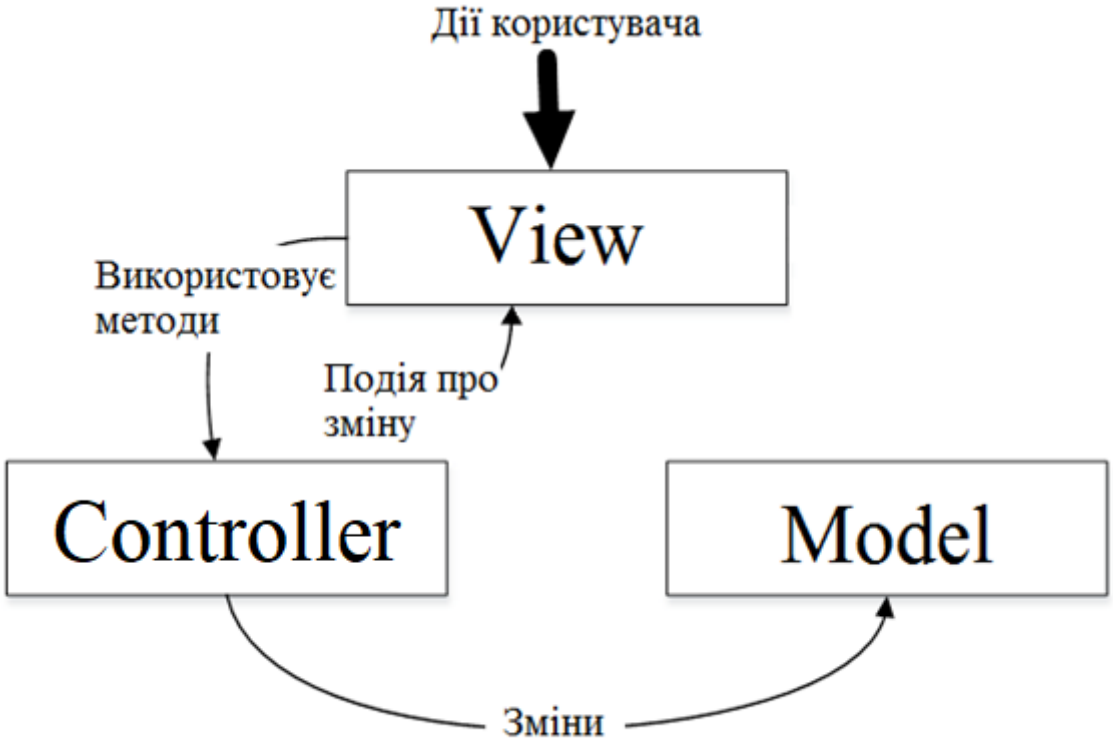

Рисунок 2.4 – Схема роботи MVC шаблону

Модель — містить знання про предметну область, дані та правила доступу до цих даних, але нічого не знає про контролери та представлення. У моделі відбувається бізнес-логіка роботи застосунку. Модель надає контролеру дані які запрошує користувач, або інша система.

Представлення — надає можливість по-різному відображати дані отримані будь-яким способом від моделі, може містити в собі логіку. Представлення — це кінцевий інтерфейс, з яким взаємодіє користувач. Користувач може передавати дані через представлення.

Контролер — реалізує взаємодію між моделлю і представленням. У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов’язаних дій в окремий клас.

Веб застосунок було реалізовано з використанням фреймворку, в основу якого було покладено архітектура Flux (рисунок 2.5).

Flux-архітектура - архітектурний підхід або набір шаблонів програмування для побудови призначеного для користувача інтерфейсу веб-додатків, що поєднується з реактивним програмуванням і побудований на односпрямованих потоках даних.

Основною відмінною рисою Flux є одностороння спрямованість передачі даних між компонентами Flux-архітектури. Архітектура накладає обмеження на потік даних, зокрема, виключаючи можливість поновлення стану компонентів самими собою. Такий підхід робить потік даних передбачуваним і дозволяє легше простежити причини можливих помилок в програмному забезпеченні а також дозволяє розробнику використовувати TTD.

Процес TTD, або налагодження подорожі в часі або налагодження під час подорожі в часі - це процес повернення назад у часі через вихідний код, щоб зрозуміти, що відбувається під час виконання комп'ютерної програми[8]. Як правило, налагодження та відладники, інструменти, які допомагають користувачеві в процесі налагодження, дозволяють користувачам призупинити виконання запущеного програмного забезпечення та перевірити поточний стан програми. [9] Потім користувачі можуть перейти вперед у часі, перейти до заяв або над ними і перейти вперед. [10] Інтерактивні відладники включають можливість модифікувати код і зробити крок вперед на основі оновленої інформації. [11] Інструменти зворотного налагодження дозволяють користувачам переходити назад у часі через дії, які призвели до досягнення певної точки програми. Відладники, що подорожують в часі, надають ці функції, а також дозволяють користувачам взаємодіяти з програмою, змінюючи історію, якщо це потрібно, і дивитися, як програма реагує [12].

Є кілька характеристик, які підтримують здатність рухатися вперед, а також вперед у часі :

- . Вибір чисто функціональної мови програмування допомагає завдяки самодостатній природі чистих функцій. Чисті функції не мають побічних ефектів і залежать тільки від інформації, явно наданої функції, забезпечуючи повторюваний, надійний, повторно відтворюваний шлях через код.

- Мови та відладники, які дозволяють гарячу заміну, можливість змінювати код під час роботи коду, забезпечують деякі вимоги, необхідні для перемотування назад, і потенційно переписування виконання. [13] [14]

У мінімальному варіанті Flux-архітектура може містити три шари, які взаємодіють один по одному:

- Actions (дії)
- Stores (сховища)
- Views (подання)

Хоча зазвичай між діями і сховищами додають Dispatcher (диспетчер).

В першу чергу Flux працює з інформаційною архітектурою, яка потім відбивається в архітектурі програмного забезпечення, тому рівень уявлень слабо зачеплений з іншими рівнями системи. [32]

Дії - вираз подій. Диспетчери передають дії нижчого компонентів (сховищ) по одному. Нова дія не передається поки попередня повністю не оброблена компонентами. Дії через роботу джерела дії, наприклад, користувача, надходять асинхронно, але їх диспетчеризація є синхронним процесом. Крім імені, дії можуть мати корисне навантаження, що містить пов'язані з дією дані. Фактично диспетчер керує всім потоком даних у програмних застосунках, які використовують Flux-архітектуру.

Дії можуть також надходити з інших місць, наприклад, з сервера. Це відбувається, наприклад, під час ініціалізації даних. Це також може статися, коли сервер повертає код помилки або коли сервер має оновлення для надання додатку [33].

Диспетчер призначений для передачі дій сховищам. У спрощеному варіанті диспетчер може взагалі не виділятися, як єдиний на весь додаток. У диспетчері сховища реєструють свої функції зворотного виклику і залежності між сховищами.

При зростанні додатку, диспетчер стає більш важливим, оскільки його можна використовувати для управління залежностями між сховищами, викликаючи зареєстровані зворотні виклики в певному порядку. Сховища можуть декларативно

чекати, поки інші сховища закінчать оновлення, а потім відповідним чином оновитись.

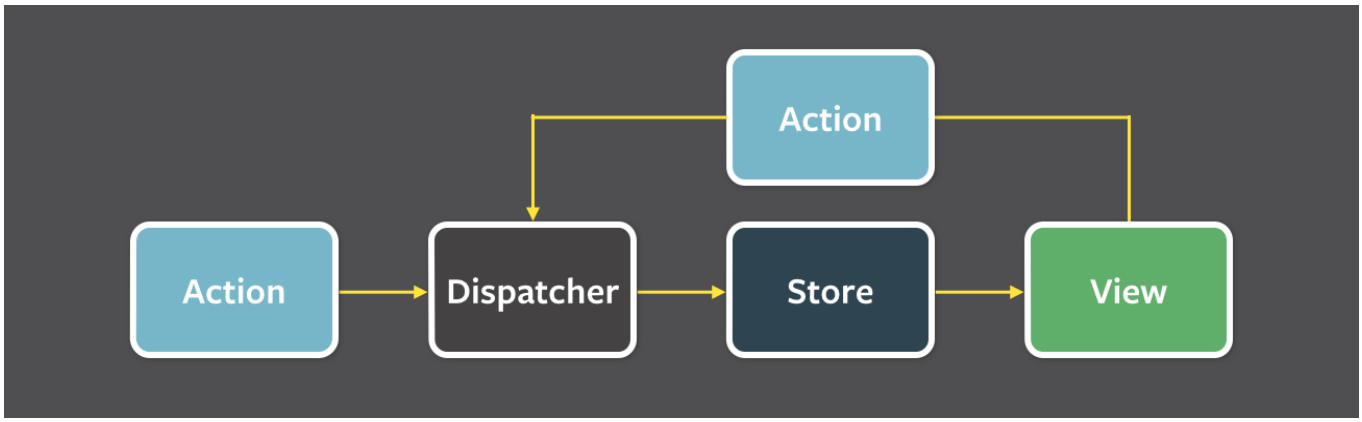

Рисунок 2.5 – Схема роботи архітектури Flux

Сховище містить стан програми та логіку. Його роль дещо схожа на модель в традиційному MVC, але воно керує станом багатьох об'єктів - воно не являє собою єдиний запис даних, як моделей ORM. Воно не є таким ж, як і колекції Backbone. Більше, ніж просто керування набором об'єктів стилю ORM, сховище керує станом програми для певного домену в межах програми.

Наприклад, редактор відеозапису Facebook використовує TimeStore, який відстежує час відтворення та стан відтворення. З іншого боку, в ImageStore того ж самого додатка зберігається колекція зображень. TodoStore в нашому прикладі TodoMVC подібний тим, що він керує колекцією завдань. Сховище демонструє характеристики як колекції моделей, так і одиночної моделі логічного домену.

Як згадувалося вище, сховище реєструється з диспетчером і надає йому зворотний виклик. Цей зворотний виклик отримує дію як параметр. В межах зареєстрованого зворотного виклику сховища, вираз комутатора, заснований на типі дії, використовується для інтерпретації дії. Це дозволяє дії призвести до оновлення стану сховища, за допомогою диспетчера. Після оновлення сховища воно транслює подію, яка оголошує, що його стан змінився, тому представлення можуть запитувати новий стан і оновлювати себе.

Представлення - компонент, який зазвичай відповідає за видачу інформації користувачеві. У Flux-архітектурі, яка може технічно не торкатися внутрішнього облаштування уявлень взагалі, це - кінцева точка потоків даних. Для інформаційної архітектури важливо тільки, що дані потрапляють в систему (тобто, назад в сховища) тільки через дії. Також представлення відповідає за введення даних користувачем і надає йому можливість викликати певні дії.

Основною метою застосування даної концепції є відділення бізнес-логіки (моделі) від її представлення. [2] Такий поділ підвищує гнучкість системи та можливість повторного використання. Для великих проектів, використання поданого шаблону дозволяє тестувати різноманітні компоненти програмної системи.

2.4 Опис інструментів розробки

Архітектура програмного комплекс складається з трьох рівнів. Кожен рівень реалізовано за допомогою різних технологій, а головною ціллю є створення мультиплатформного рішення на базі відкритих технологій.

Технології, що використовуються для клієнтського рівня: мова програмування JavaScript, фреймворк для побудови веб-застосунків React.js, фреймворк для створення графічних інтерфейсів Antd.

JavaScript — мова програмування, що поєднує в собі декілька стилів програмування — об'єктно-орієнтований, імперативний і функціональний[6]. Є реалізацією мови ECMAScript. Найбільш широке застосування знаходить в браузерах як мова сценаріїв для додання інтерактивності веб-сторінок.

Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам'яттю, прототипне програмування, функції як об'єкти першого класу.

На JavaScript вплинули багато мов, при розробці була мета зробити мову схожим на Java, але при цьому легким для використання непрограмістів. Мовою

JavaScript не володіє будь-яка компанія або організація, що відрізняє його від ряду мов програмування, використовуваних в веб-розробці.

React.js — JavaScript-бібліотека для створення користувацьких інтерфейсів. React — це декларативна, ефективна і гнучка JavaScript-бібліотека, призначена для створення інтерфейсів користувача, що використовує односпрямований потік даних.. Вона дозволяє компонувати складні інтерфейси з невеликих окремих частин коду — “компонентів”. [19]

React використовує той факт, що логіка виводу пов’язана з іншою логікою інтерфейсу користувача: як обробляються події, як змінюється стан з часом і як дані готуються для рендерингу. [30]

Замість того, щоб штучно відокремити технології, розмістивши розмітку і логіку в окремих файлах, React розділяє відповідальність між вільно зв’язаними одиницями, що містять обидві технології і називаються “компонентами” [4][31].

Redux — шаблон для JavaScript з відкритим вихідним кодом, призначений для управління станом додатку. Найчастіше використовується в зв'язці з React або Angular для розробки клієнтської частини. Містить ряд інструментів, що дозволяють значно спростити передачу даних сховища через контекст. [5] Творці: Ден Абрамов і Ендрю Кларк.[18]

Redux – шаблон з простим API, передбачуваним сховищем стану додатку. Він працює по тому ж самому принципу, що і редюсер, один з концептів функціонального програмування. [15] Його творці були натхненні функціональною мовою програмування Elm.

Шаблон Redux допомагає писати програми, які ведуть себе послідовно, запускаються в різних середовищах (клієнт, сервер), і їх легко тестувати. Крім того, він забезпечує великий досвід розробників, наприклад, редагування живого коду в поєднанні з відладником у часі.

Antd — це набір інструментів, призначений для створення веб-додатків, який включає в собі шаблони CSS та HTML для типографіки, форм, кнопок, навігації та

інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-додатків. [17]

Технології, що використовуються для серверної частини: мова програмування Javascript та платформа Node.js, фреймворк для створення RESTful сервісів Express.js, ORM для доступу до бази даних mongodb.js.

Node.js — платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Засновником платформи є Раян Дал (Ryan Dahl). Якщо раніше Javascript застосовувався для обробки даних в браузері на сторонні користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їх виконання. [19] Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників [29].

Node.js має наступні властивості:

- асинхронна однопотокова модель виконання запитів;
- неблокуючий ввід/вивід;
- система модулів CommonJS;
- рушій JavaScript Google V8;

Для керування модулями використовується пакетний менеджер npm (node package manager).

Платформа окрім роботи із серверними скриптами для веб-запитів, також використовується для створення клієнтських та серверних програм.

В платформі використовується розроблений компанією Google рушій V8.

Для забезпечення обробки великої кількості паралельних запитів у Node.js використовується асинхронна модель запуску коду, заснована на обробці подій в неблокуючому режимі та визначенні обробників зворотних викликів (callback). Як способи мультиплексування з'єднань підтримується epoll, kqueue, /dev/poll і select.

Для мультиплексування з'єднань використовується бібліотека libuv, для створення пулу потоків (thread pool) задіяна бібліотека libeio, для виконання DNS-запитів у неблокуючому режимі інтегрований c-ares. Всі системні виклики, що

спричиняють блокування, виконуються всередині пулу потоків і потім, як і обробники сигналів, передають результат своєї роботи назад через неіменовані канали (pipe).

За своєю суттю Node.js схожий на фреймворки Perl AnyEvent, Ruby Event Machine і Python Twisted, але цикл обробки подій (event loop) у Node.js прихований від розробника і нагадує обробку подій у веб-застосунку, що працює в браузері.

Express.js — це мінімалістичний і гнучкий веб-фреймворк для додатків Node.js, що надає великий набір функцій для мобільних і веб-додатків. [21]

Express.js реалізований як вільне і відкрите програмне забезпечення під ліцензією MIT. Де-факто є стандартним каркасом для Node.js. Автор фреймворка, TJ Holowaychuk, описує його як створений на основі написаного на мові Ruby каркаса Sinatra, маючи на увазі, що він мінімалістичний і включає велику кількість додаткових плагінів. Express може бути сервером для програмного стека MEAN, разом з базою даних MongoDB і каркасом React.js.

Mongodb.js — це крос-платформний фреймворк для забезпечення уніфікованого доступу до MongoDB. [22][28]

Для рівня бази даних було обрано MongoDB.

MongoDB — документо-орієнтована система керування базами даних (СКБД) з відкритим вихідним кодом, яка не потребує опису схеми таблиць [7]. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів. [26]

Код MongoDB написаний на мові C++ і поширюється в рамках ліцензії AGPLv3. [23]

MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову

відмовостійких конфігурацій. [24] У MongoDB є вбудовані засоби із забезпечення шардінгу (розподіл набору даних по серверах на основі певного ключа), комбінуючи який з реплікацією даних можна побудувати горизонтально масштабований кластер зберігання, в якому відсутня єдина точка відмови (збій будь-якого вузла не позначається на роботі БД), підтримується автоматичне відновлення після збою і перенесення навантаження з вузла, який вийшов з ладу. Розширення кластера або перетворення одного сервера на кластер проводиться без зупинки роботи БД простим додаванням нових машин. [25][27]

2.5 Обґрунтування вибору програмної реалізації

Під час проектування системи було проведено аналіз предметної області та вимог замовника. Було прийняте рішення розроблювати програмний продукт, який заснований на веб-технологіях та мобільний додаток.

Технології, що були використані на сервері, відповідають принципам відкритості вихідних кодів, актуальності в наш час та можливістю виконання на будь-якій операційній системі. Тому було обрано Express.js, який дозволяє швидко та якісно розробити як великі серверні системи, так і маленькі. Також цей фреймворк дозволяє використовувати безмежну кількість різноманітних пакетів, які підтримуються Node.js, які чудово працюють з цим фреймворком. Express.js дозволяє дуже швидко зробити робочу версію програми і встановити її на будь-якій машині. Для доступу до бази даних було обрано пакет MongoClient, який дозволяє швидко та дуже просто зробити підключення до бази даних та має дуже простий і зрозумілий API інтерфейс. Для управлінням пакетами було використано npm – застосунок від розробників Node.js, який дозволяє дуже просто управляти пакетами та бібліотеками, які застосовуються у додатку.

На клієнтському рівні було обрано технології, які задовольняють такі ж умови як і серверні, але з поправкою на виконання в браузері. Фреймворк React.js було обрано для створення складних графічних інтерфейсів, які легко тестувати та розширювати при подальшій розробці програмного забезпечення. Мова програмування Javascript була обрана через те, що вона забезпечує несувору типізацію та підтримує систему модулів. Подана мова програмування надає різноманітний “синтаксичний цукор”, що дозволяє скоротити об’єм шаблонного коду. Для впровадження архітектури Flux, була використана бібліотека Redux, яка надає змогу повністю впровадити цю архітектуру у фреймворк React.js. Фреймворк Antd було обрано для стилізації через його бібліотеку інтерфейсів та вбудовану підтримку розмірів мобільного телефону. Для того, щоб робити як тестові так і робочі версії веб застосунку.

Базою даних було обрано MongoDB через відкритий вихідний код і великою підтримкою з боку розробників. Обрану базу даних можна розгорнути як в Docker контейнері, так і на будь-якій операційній системі.

Сукупність обраних технологій дає змогу побудувати надійний і якісний програмний продукт, що захищено від патентних позовів, так як всі використані технології покриті ліцензіями, що дають доступ до вихідного коду.

2.6 Висновки до розділу

У розділі “ Засоби реалізації програмної ” було описано мову програмування Javascript. Наведено список її особливостей та переваг.

Розглянуто використані при розробці платформи, бібліотеки функцій. Наведено сфери їх застосування. Надано опис Flux архітектури, бібліотек React.js, Redux.js, Express.js, за допомогою яких було реалізовано графічний інтерфейс демонстраційного додатку.

3. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Освітня платформа буде складатися з чотирьох модулів, розбитих на різну кількість функціональних під блоків. Головним модулем є кабінет користувача, що одночасно є елементом компонування системи. Обрана структура дозволить інтуїтивно користуватися системою за рахунок розташування модулів у порядку, що зазвичай використовують для вирішення схожих задач.

3.1 Опис функціональності системи

Програмний застосунок для освітньої web-система містить у собі двох головних акторів – користувач системи і адміністратор системи; Головна відмінність у тому, що адміністратор має можливість створення курсів через спеціальну сторінку програмного застосунку.

3.1.1 Діаграма прецедентів

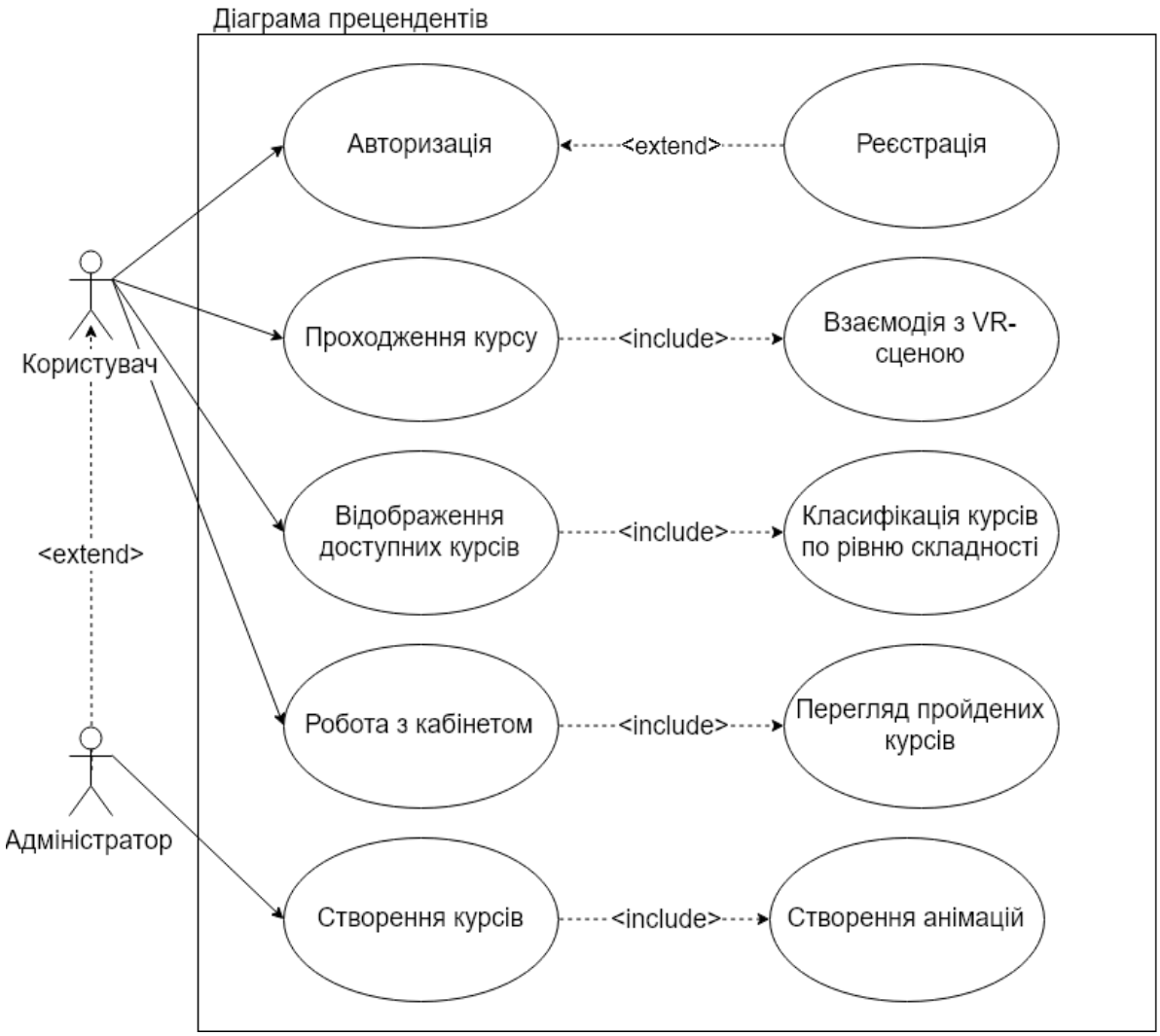

Рисунок 3.1 – Діаграма прецедентів

На рисунку 3.1 зображена діаграма, на якій продемонстровані основні актори системи — Користувач і Адміністратор.

3.2 Концептуальна модель бази даних

База даних складається з дванадцяти взаємопов’язаних таблиць, які створюють інформаційний простір для зберігання та отримання доступу до даних.

Головною таблицею є “Користувач”, де міститься основна інформація про користувача, та про пройдені ним курси.

Таблиця “Курс” містить основну інформацію про кожен конкретний курс, його назву, опис тощо.

Таблиця “Урок” містить інформацію про конкретний урок, список його етапів, приблизна оцінка часу на проходження тощо.

Таблиця “Крок” містить інформацію про конкретний етап уроку: заголовок, короткий опис етапу, типу етапу, приблизна оцінка часу на проходження тощо.

Таблиця “Сенсори” містить інформацію про всі доступні сенсори.

3.3 Опис таблиць бази даних

Для кожної таблиці створено клас на мові Javascript, що є відображенням таблиці з бази даних в об’єктній формі і використовується для доступу до даних.

Для зручності отримання даних з таблиць, що мають зв’язок типу “один-до-багатьох” та “багато-до-багатьох” було використано об'єктно-реляційної проекції.

База даних реалізована за допомогою ORM mongodb.js. Розглянемо більш детально структури кожної із таблиць бази даних системи освітньої платформи.

У таблицях 3.1-3.5 наведена детальна інформація про структури таблиць бази.

Таблиця 3.1. Структура таблиці “Користувач”

Ім’я поля	**Тип поля**	**Опис поля**
_id	Mongo.ObjectId	Первинний ключ
name	String	Ім’я користувача
email	String	Адреса електронної пошти
password	String	Хеш паролю

avatar	Number	Номер аватара з представлених
completedLessons	Array<Mongo.ObjectId>	Масив первинних ключів для таблиці lessons
completedCourses	Array<Mongo.ObjectId>	Масив первинних ключів для таблиці courses
lastLesson	Mongo.ObjectId	Первинний ключ останнього уроку, який відвідав користувач
lastCourse	Mongo.ObjectId	Первинний ключ останнього курсу, який відвідав користувач
lastStep	Number	Номер останнього кроку, на який відвідав користувач
about	String	Опис користувача

Таблиця 3.2. Структура таблиці “Курс”

Ім’я поля	**Тип поля**	**Опис поля**
_id	Mongo.ObjectId	Первинний ключ
title	String	Ім’я курсу
category	Array<string>	Список категорій
description	String	Опис курсу
tags	Array<Mongo.ObjectId>	Список тегів
status	String	Статус курсу
lastUpdate	number	Час останнього оновлення даних курсу
sections	Array<Mongo.ObjectId>	Список секцій курсу
author	Mongo.ObjectId	Первинний ключ автора курсу
price	Number	Ціна курсу

review	Array<Mongo.ObjectId>	Список первинних ключів відгуків

Таблиця 3.3. Структура таблиці “Урок”

Ім’я поля	**Тип поля**	**Опис поля**
_id	Mongo.ObjectId	Первинний ключ
title	String	Ім’я уроку
timeEstimate	Number	Приблизна оцінка часу проходження уроку
steps	Array<Mongo.ObjectId>	Список первинних ключів Етапів

Таблиця 3.4. Структура таблиці “Етап”

Ім’я поля	**Тип поля**	**Опис поля**
_id	Mongo.ObjectId	Первинний ключ
title	String	Ім’я етапу
description	String	Опис етапу
content	String	Контент етапу
line	String	Опис курсу
timeEstimate	Number	Приблизна оцінка часу проходження етапу
type	Enum	Тип етапу

Таблиця 3.5. Структура таблиці “Сенсори”

Ім’я поля	**Тип поля**	**Опис поля**
_id	Mongo.ObjectId	Первинний ключ
type	String	Ім’я сенсору
index	Number	Порядковий номер сенсору
pins	Array<Mongo.ObjectId>	Список доступних пінів сенсору

3.4 Розробка кабінету користувача

Кабінет користувача є основним робочим місцем користувача в системі.

Кабінет користувача включає в себе інші підмодулі, що виконують основні функції системи.

Під модулі кабінету користувача:

— модуль відображення списку курсів;

— модуль відображення уроку;

— модуль створення уроку.

3.4.1 Модуль відображення списку курсів

Даний модуль дозволяє користувачу додатку вибрати курс зі списку:

— всіх доступних курсів;

— курсів, рекомендованих для проходження;

— курсів, розділених по складності.

3.4.2 Модуль відображення уроку

Даний модуль виконує всю роботу пов’язану з проходженням уроку, а саме:

— відображення орієнтовного часу на завершення уроку;

— відображення поточної задачі;

— прогрес по уроку.

3.4.3 Модуль створення уроку

Даний модуль виконує всю роботу пов’язану з створенням уроку, а саме:

— вибору версії Arduino;

— вибір сенсорів і їх кількості, необхідних для анімації;

— розміщення сенсорів на сцені;

— додавання тесту\коду\посилання, залежно від типу етапу;

— комбінування етапів в урок.

3.5 Висновки до розділу

У цьому розділі було розглянуто принципи реалізації освітньої web-платформи. Описано основні програмні модулі та їх взаємодії.

Розглянуті основні актори системи, наведено діаграму прецедентів.

4. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Розроблений програмний комплекс працює в будь-яких браузерах, що підтримують актуальні веб-стандарти.

Мобільний застосунок встановлюється на актуальні версії операційних систем IOS та Android.

4.1 Інсталяція та системні вимоги

Оскільки веб-застосунок працює з використанням веб-технологій, тому він не потребує встановлення на пристрій користувача. Проте необхідний веб-браузер, що підтримує актуальні веб-стандарти.

Для використання мобільного застосунку потрібно пристрій із Android версії від 4.4 або IOS від 9 версії. Також потрібно мінімум 2 Гб оперативної пам’яті на пристрої та 100 мб на носії.

4.2 Інструкція з використання програмного продукту

При вході на веб додаток, користувач має авторизуватися в системі, для того щоб почати роботу в системі. На рисунку 4.1 зображена форма авторизації.

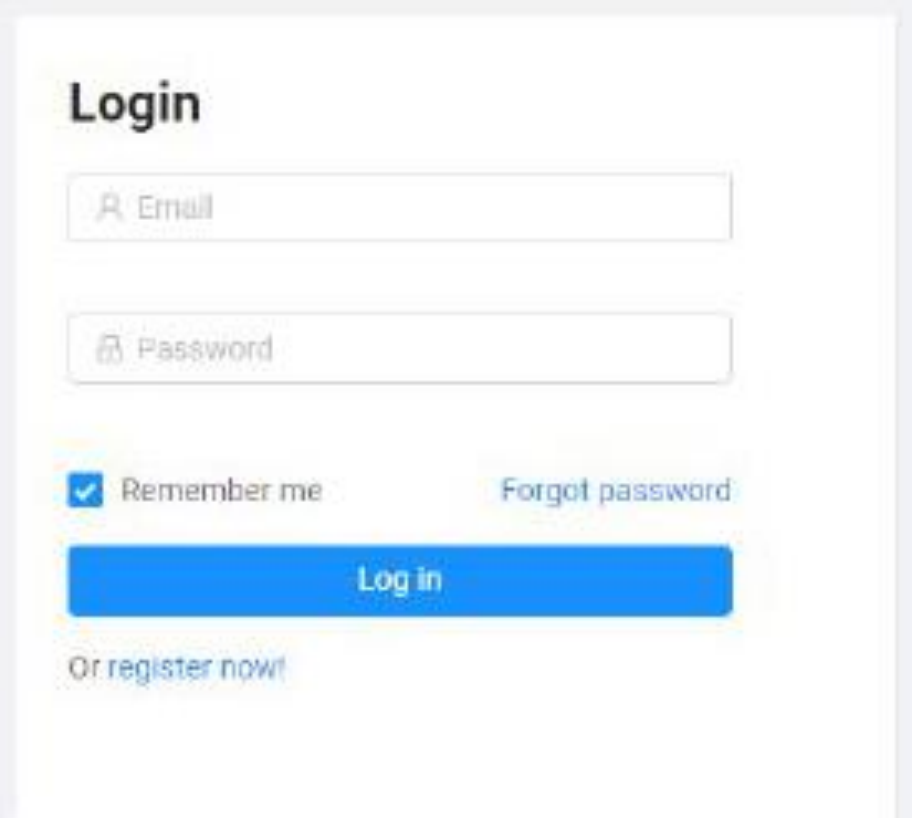

Рисунок 4.1 — Форма авторизації

Якщо користувач вперше користується системою, йому необхідно зареєструватися в системі. На рисунку 4.2 зображена форма реєстрації.

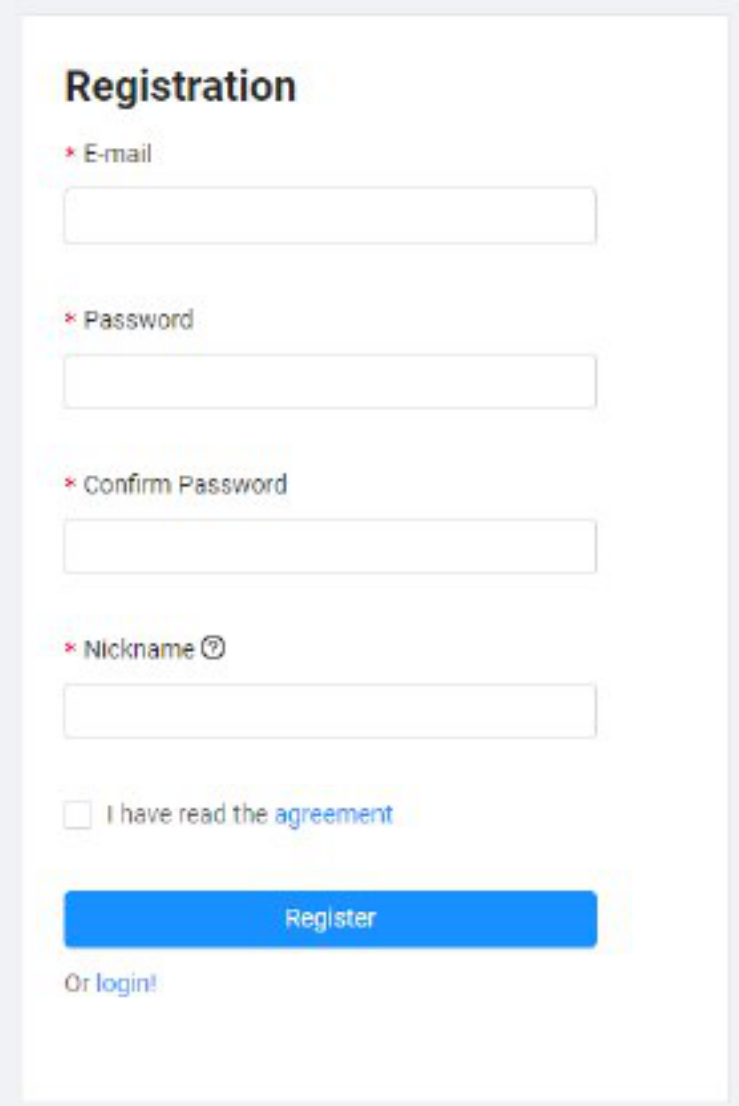

Рисунок 4.2 — Форма реєстрації

Після авторизації в системі, користувач отримує доступ до головного меню. За допомогою нього він отримує доступ до усіх сторінок системи, а також має можливість вийти зі свого профілю.

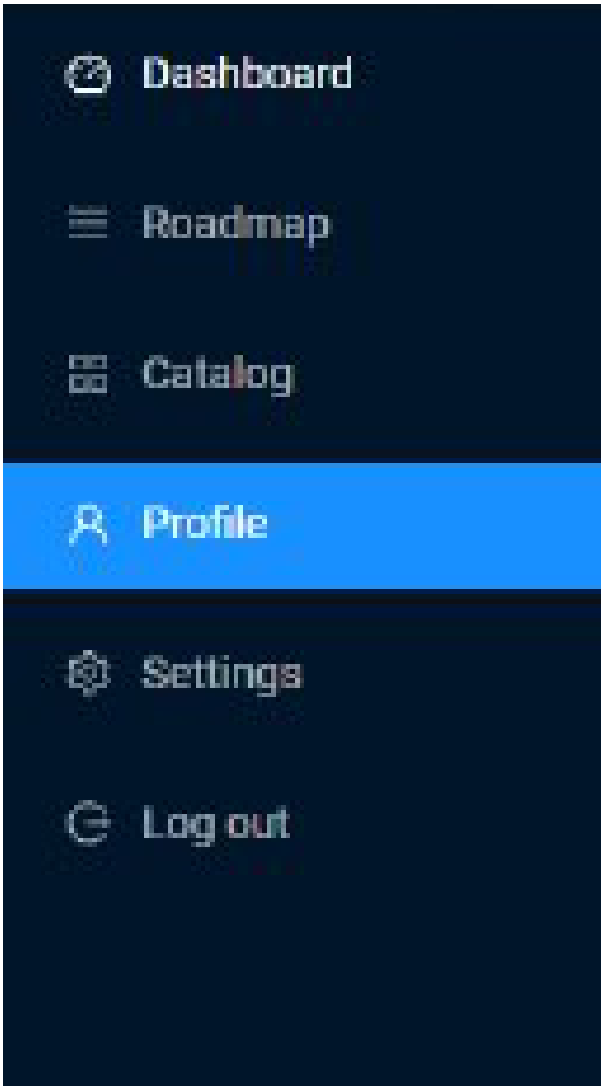

Рисунок 4.3 — Головне меню системи

Також він може перейти до свого основного робочого простору – кабінету користувача. На рисунку 4.3 зображене головне меню системи.

Одним із важливих компонентів головного меню є сторінка вибору курсів. За її допомогою користувач має змогу обрати, переглянути або зарахуватися на курс. На рисунку 4.4 зображене меню вибору курсів.

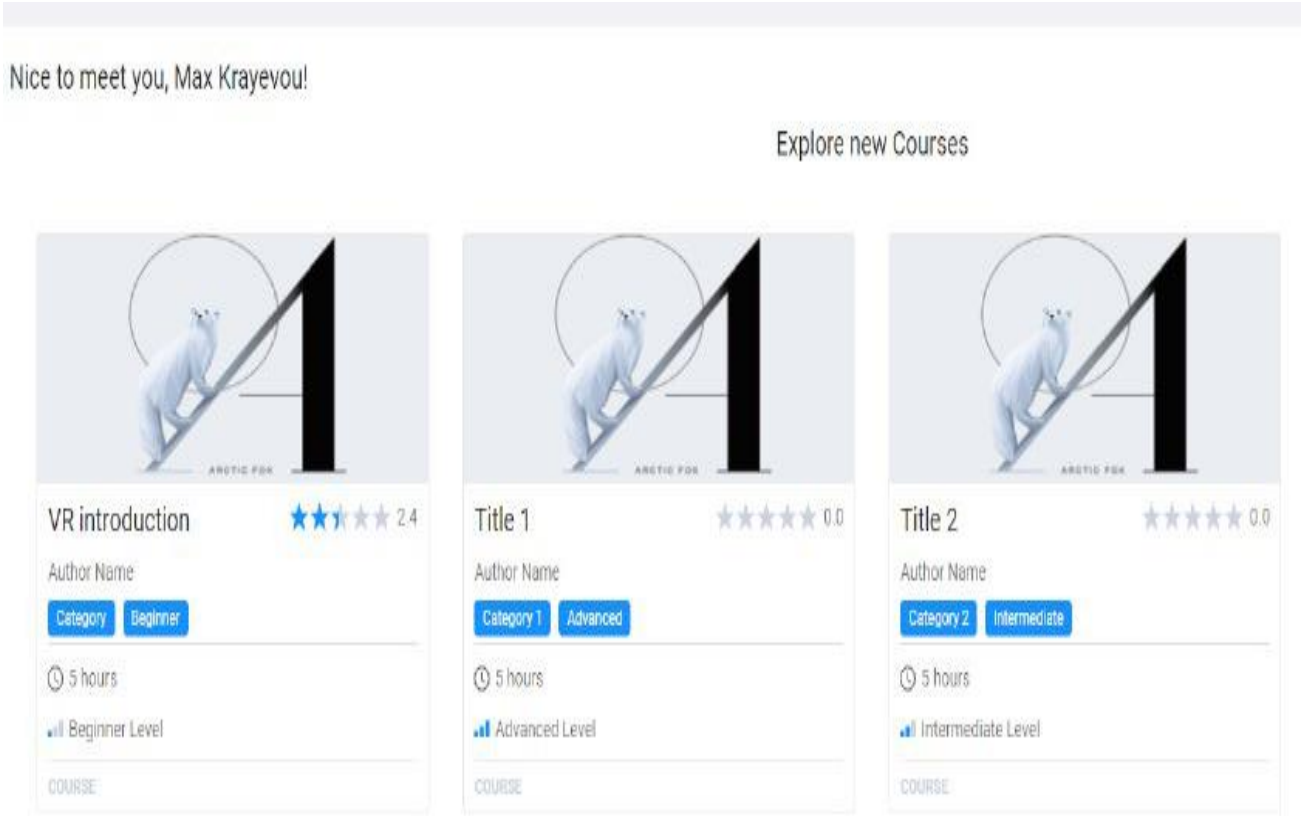

Рисунок 4.4 — Меню вибору курсів

Кожен курс може складатися із багатьох уроків. Кожен урок можна пройти, у ньому можуть бути різноманітні види навчання. Після проходження уроку, користувачеві відкривається наступний. Після проходження останнього уроку курсу, йому показується повідомлення про успішне закінчення курсу.

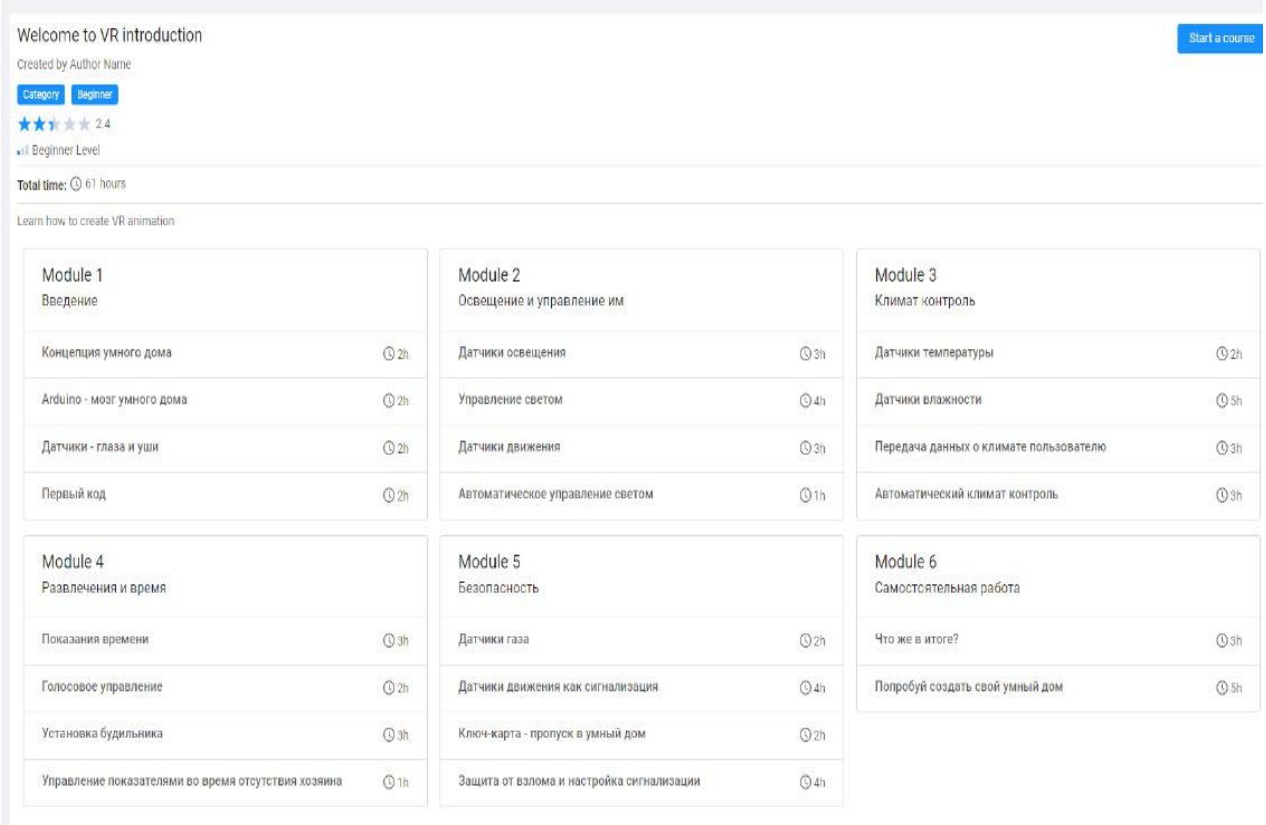

Рисунок 4.5 — Меню курсу із уроками

На рисунку 4.5 зображене меню курсу з уроками.

Курс розділено на модулі за темами, кожен модуль має орієнтовну оцінку по часу на його проходження.

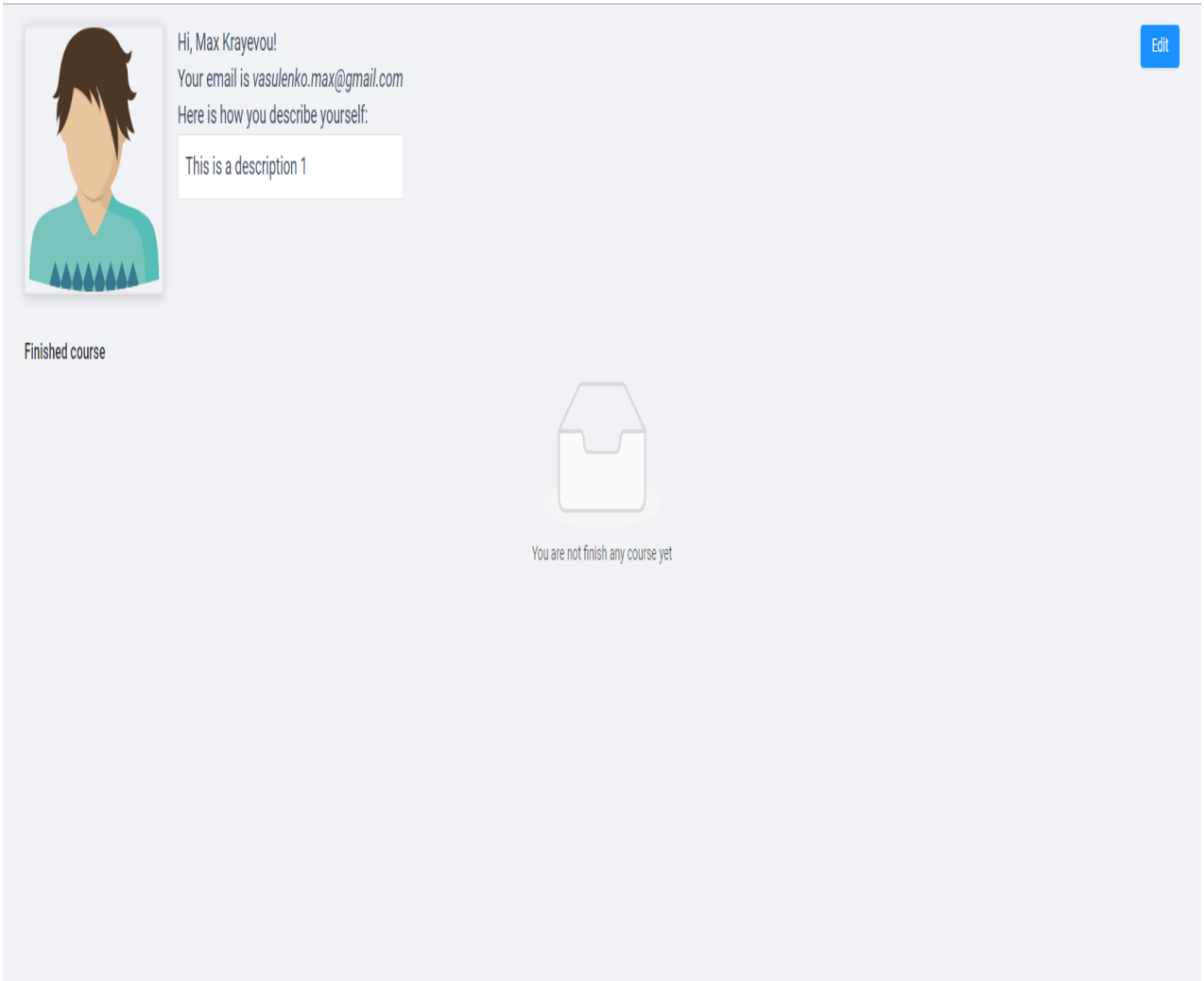

Рисунок 4.6 — Особистий кабінет користувача

В особистому кабінеті користувач може змінити свої особисті дані, а також проглянути прогрес по курсам.

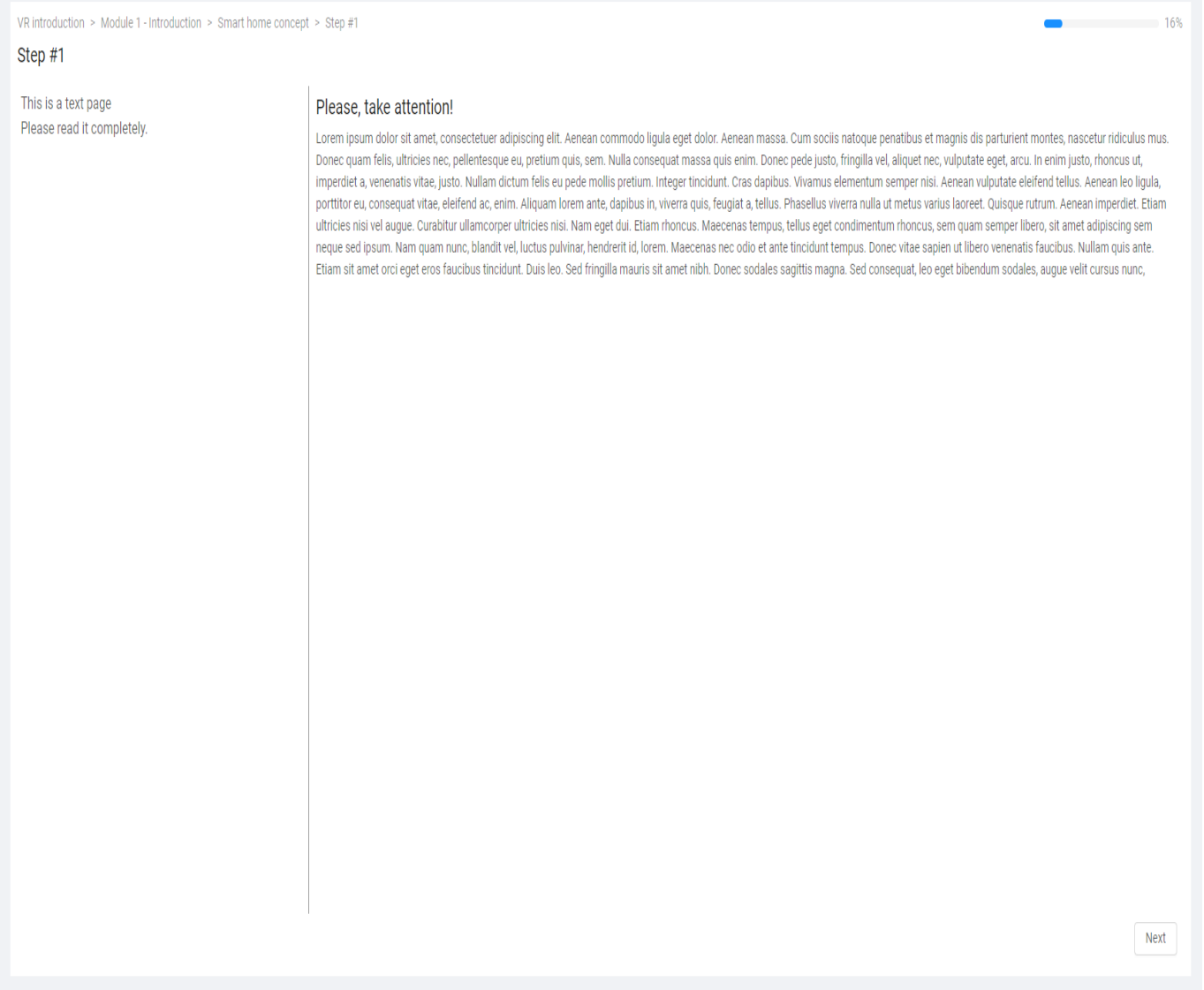

Рисунок 4.7 — Сторінка етапу

Кожен етап містить в собі короткий опис етапу, його ім’я, орієнтовний час на його завершення. Також на сторінці етапу можна отримати інформацію по прогресу по курсу.

Етапи можуть бути наступних типів:

- Теоретичний
- Практичний
- Відео
- Тестовий
- VR-сцена

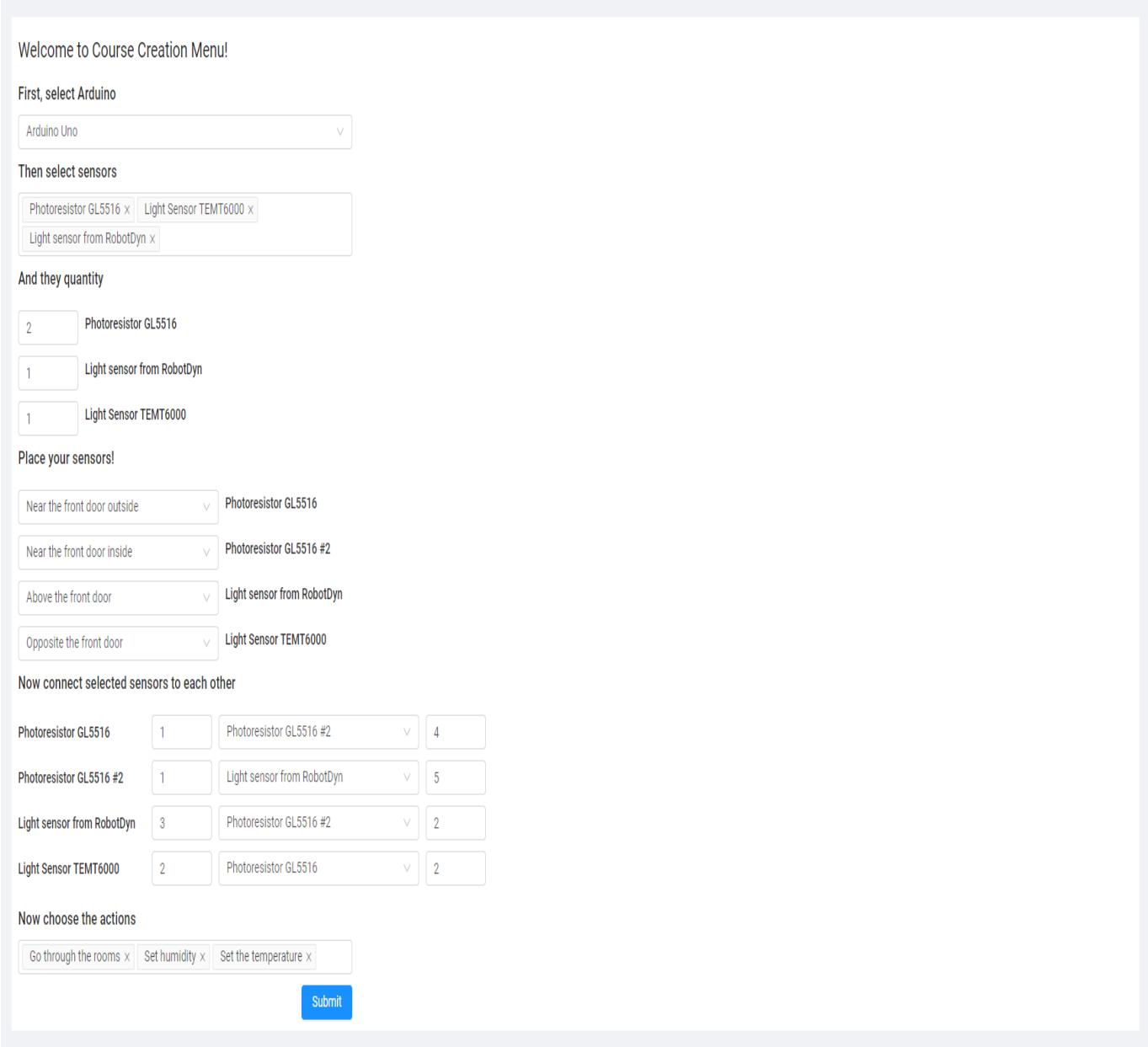

Рисунок 4.8 — Сторінка створення анімацій для курсу

Для створення анімації необхідно обрати версію Arduino, вибрати типи сенсорів і їх кількості, необхідних для анімації, розмістити сенсори на сцені, поєднати сенсори між собою і обрати дії, які має зробити актор для перевірки коректності роботи анімації.

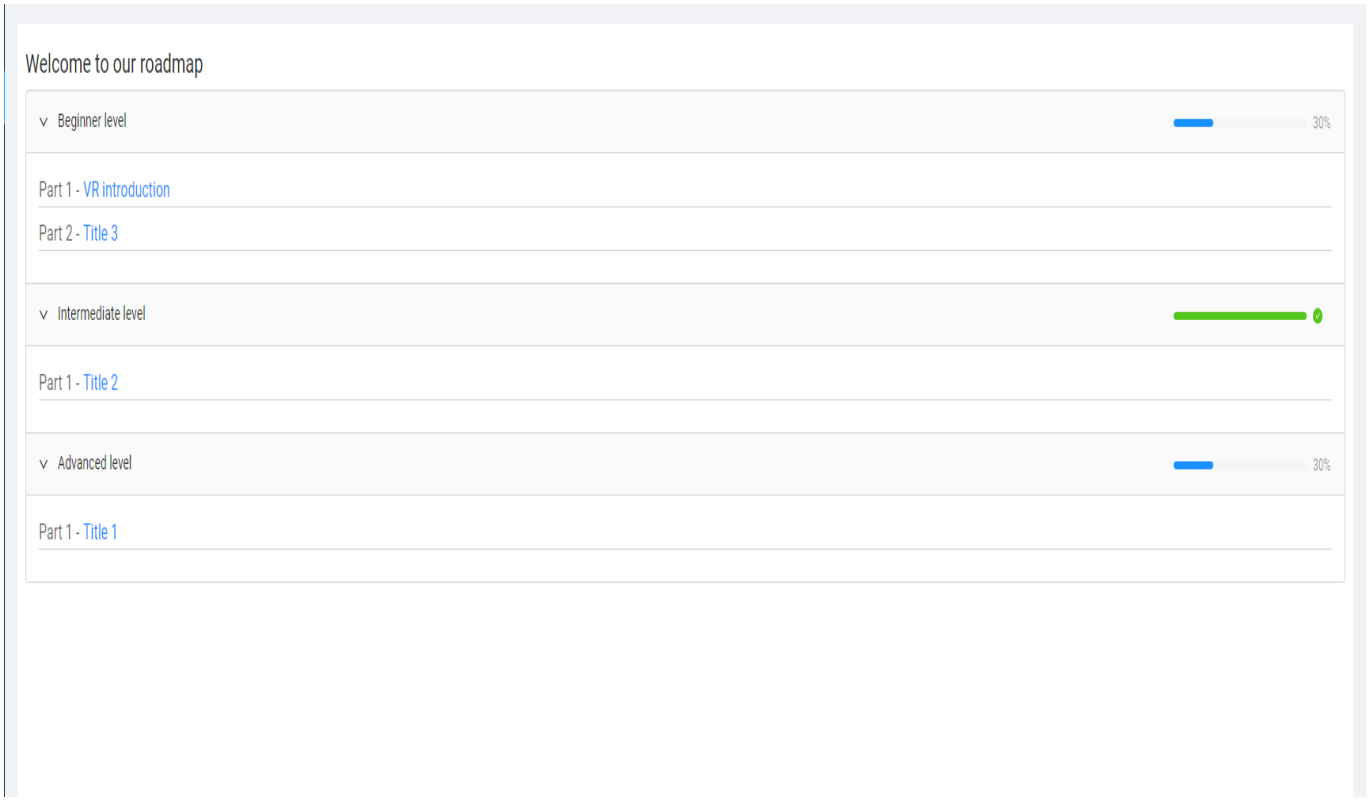

Рисунок 4.9 — Сторінка з рекомендованими курсами

Рекомендація по курсам базується на доступних курсів, відсортованих за складністю.

4.3 Висновки до розділу

У цьому розділі було надано опис процесу роботи користувача з розробленою платформою.

Надано опис основних інтерфейсів і описано їх функціональну частину.

Надано інформацію про принципи роботи демонстраційного додатку.

Продемонстровано роботу користувача з графічним інтерфейсом демонстраційного додатку.

ВИСНОВКИ

У ході аналізу існуючого сучасних освітніх платформ було досліджено системи, які слугують для вирішення поставлених задач. При аналіз було показано, що існуючі системи не дають повного поглинання у вивчений матеріал, а отже вирішують задачу не у повному обсязі,.

Розроблений програмний продукт дозволяє створювати курси для вивчення процесі розумних будинків, а також проходити їх, залучаючи віртуальну реальність.

Проведено огляд методів і засобів розробки програмної системи. Обґрунтовано вибір створення програмної системи, заснованої на веб-технологіях. Клієнт-серверна архітектура додатку дає змогу підвищити гнучкість системи, та зручність у розробці та супроводі.

Користувачами системи можуть бути різноманітні користувачі, які хочуть вивчити як будувати та програмувати розумні будинки і мають базові навички програмування. Програмне забезпечення може бути використано на будь-якому сучасному смартфоні, а також на будь-якій операційній системі, де встановлено браузер з підтримкою останніх веб-стандарти, а також має доступ до інтернету.

За результатами виконання тестових завдань підтверджена коректність отриманих результатів, отже система відповідає поставленим вимогам.

Досліджено різноманітні техніки та алгоритми побудови програмного забезпечення, було виявлено різноманітні сфери, для яких можна застосовувати дані прийоми. В основу розробленого програмного забезпечення лягли створені прототипи програмного забезпечення, які вирішували різні аспекти поставленої задачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Адам Фримен — ASP.NET Core MVC с примерами на C# для профессионалов [Електронний ресурс]. — 2017. — Режим доступу: https://bit.ly/2JbSgHe.

2. Джеймс Чамберс — ASP.NET Core. Разработка приложений [Електронний ресурс]. — 2018. — Режим доступу: https://bit.ly/2AliYL2.

3. Markus Egger — MVVM Survival Guide for Enterprise Architectures in Silverlight and WPF [Електронний ресурс]. — 2012. — Режим доступу: https://www.packtpub.com/application-development/mvvm-survival-guide-enterprise-architectures-silverlight-and-wpf.

4. Офіційний сайт бібліотеки React.js [Електронний ресурс]– Режим доступу: https://uk.reactjs.org/.

5. Anthony Gore — Full-Stack Javascript [Електронний ресурс]. — 2015. – Режим доступу: https://bit.ly/2OEODzR.

6. Kyle Simpson - You Don't Know JS: Up & Going [Електронний ресурс] -2015. - Режим доступу: https://www.ebooks.com/en-ua/1993212/you-don-t-know-js-up-going/simpson-kyle.

7. Болье А. — Learning NoSQL [Електронний ресурс]. — 2005. — Режим доступу: http://shop.oreilly.com/product/9780596007270.do.

8. Telles, Matthew; Hsieh, Yuan (2001-04-01). The Science of Debugging. Coriolis Group Books.

9. Time Travel Debugging in WinDbg Preview!". Debugging Tools for Windows. Retrieved 2018-05-08

10. "Reverse debugging, time travel debugging". undo.io. Retrieved 2018-05-08.

11. "Interactive Debugging With Node.js — DZone Web Dev". dzone.com. Retrieved 2018-05-08.

12. "Elm's Time Travelling Debugger". debug.elm-lang.org. Retrieved 2018-05-08.

13 "Interactive programming". elm-lang.org. Retrieved 2018-05-08.

14. "Hot reloading and time travel debugging: what are they?". Code Cartoons. 2015-10-21. Retrieved 2018-05-08.

15. В чём сила Redux?. habr.com. Дата обращения 11 февраля 2019.

16. Паттерн Chain of Responsibility (цепочка обязанностей) — назначение, описание, особенности и реализация на C++.

17. Офіційний сайт бібліотеки AntDesign [Електронний ресурс]– Режим доступу: https://ant.design/

18. Офіційний сайт бібліотеки Redux.js [Електронний ресурс]– Режим доступу: https://redux.js.org/

19. Офіційний сайт бібліотеки React.js [Електронний ресурс]– Режим доступу: https://reactjs.org/

20. Офіційний сайт Node.js [Електронний ресурс]– Режим доступу: https://nodejs.org/uk/

21. Офіційний сайт Express.js [Електронний ресурс]– Режим доступу: https://expressjs.com/ru/

22. Офіційний сайт MongoDB [Електронний ресурс]– Режим доступу https://www.mongodb.com/

23. Core Server Versions. MongoDB.

24. MongoDB. GridFS article on MongoDB Developer's Manual. MongoDB.

25. Кайл Бэнкер — MongoDB in Action. — ДМК Пресс, 2014.

26. Kristina Chodorow — MongoDB: The Definitive Guide, 2nd Edition. — O’Reilly, 2013.

27. Mitch Pirtle — MongoDB for Web Development. — Addison-Wesley Professional, 2011

28. Steve Hoberman — Data Modeling for MongoDB. — Technics Publications, 2014.

29. Итан Браун — Web Development with Node and Express — Санкт-Петербург: Питер, 2017.

30. Dawson, Chris — JavaScript’s History and How it Led To ReactJS. The New Stack

31. Hemel Zef — Facebook's React JavaScript User Interfaces Library Receives Mixed Reviews. InfoQ

32. Jonathan Hayward; Artemij Fedosejev; Narayan Prusty; Adam Horton; Ryan Vice; Ethan Holmes; Tom Bray. React: Building Modern Web Applications. — Packt Publishing, 2016.

33. Sandeep Kumar Patel. The flux architecture // Learning Web Component Development. — Packt Publishing, 2015.

34. LaLonde, W.R. and J.R. Pugh, Inside Smalltalk, Volume II, Prentice-Hall, 1991.

ДОДАТОК А

Освітня web-система моделювання процесів створення розумного будинку з елементами віртуальної реальності

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВз5104_18Б

Аркушів 3

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ-з5104-18Б_	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ-з5104-18Б-12_1	Src/index.js	Корневий компонент, що містить точку входа
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ-з5104-18Б-12_1	Src/App.js	Головний компонент для React.js
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ-з5104-18Б-12_1	Src/ registerServiceWorker.js	Компонент, що реалізує кешування
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ-з5104-18Б-12_1	Src/view/UserPage/index.js	Інтерфейс для сторінки особистого кабінету користувача
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ-з5104-18Б-12_1	Src/view/Roadmap/index.js	Інтерфейс для сторінки з рекомендованими курсами
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ-з5104-18Б-12_1	Src/view/Registration/index.js	Інтерфейс для сторінки реєстрації користувача

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ-з5104-18Б-12_1	Src/view/Login/index.js	Інтерфейс для сторінки аутентифікації користувача
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ-з5104-18Б-12_1	Src/view/LessonPage/index.js	Інтерфейс для сторінки уроку
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ-з5104-18Б-12_1	Src/view/Dashboard/index.js	Інтерфейс для сторінки з переліком курсів
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ-з5104-18Б-12_1	Src/view/CreateCoursePage/index.js	Інтерфейс для сторінки створення анімації
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ-з5104-18Б-12_1	Src/view/CoursePage/index.js	Інтерфейс для сторінки курсу

ДОДАТОК Б

Освітня web-система моделювання процесів створення розумного будинку з елементами віртуальної реальності

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВз5104_18Б 12-1

Аркушів 14

2019

src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import 'antd/dist/antd.css';
import './normalize.css';
import { Provider } from 'react-redux';
import App from './view/App';
import registerServiceWorker from './registerServiceWorker';
import store from './store';

ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>, document.getElementById('root'),
);
registerServiceWorker();
```

src/App.js

```
import React, { Component } from 'react';
import { connect } from 'react-redux';
import { history } from 'store';
import { bindActionCreators } from 'redux';
import { Redirect, Switch, Route } from 'react-router-dom';
import { ConnectedRouter } from 'connected-react-router';
import { /* Header, */ Sider, Spinner } from 'components';
import { Layout } from 'antd';
import 'assets/fonts/fonts.css';
import { checkIsUserLoggedIn } from '../redux/user/actions';
import Dashboard from './Dashboard';
import Login from './Login';
import Registration from './Registration';
import { getCourseList, getRoadmap } from '../redux/courses/actions';
```

```
import Roadmap from './Roadmap';
import UserPage from './UserPage';
import CoursePage from './CoursePage';
import CoursesList from './CoursesList';
import LessonPage from './LessonPage';
import CreateCoursePage from './CreateCoursePage';

const ProtectedRoute = ({
  isUserLoggedIn, path, component, exact,
}) => (
  isUserLoggedIn
    ? <Route exact={exact} path={path} component={component} />
    : <Redirect to="/login" />
);

const MainLayout = () => (
  <Layout>
    <Sider />
    <Layout>
      {/* <Header style={{ background: '#fff', padding: 0 }} /> */}
      <Layout.Content style={{ margin: '0 16px' }}>
        <Switch>
          <Route exact path="/" component={Dashboard} />
          <Route path="/courses" component={CoursesList} />
          <Route path="/roadmap" component={Roadmap} />
          <Route path="/course/:id/lesson/:lessonId" component={LessonPage} />
          <Route path="/course/:id" component={CoursePage} />
          <Route path="/create-course" component={CreateCoursePage} />
          <Route path="/profile" component={UserPage} />
          <Redirect to="/" />
        </Switch>
      </Layout.Content>
    </Layout>
  </Layout>
```

```
);

class App extends Component {
  constructor(props) {
    super(props);
    this.state = {};
    this.initReduxPart();
  }

  initReduxPart = () => {
    const { actions: { checkIsUserLoggedIn, getCourseList, getRoadmap } } = this.props;
    checkIsUserLoggedIn();
    getCourseList();
    getRoadmap();
  };

  render() {
    const { isUserLoggedIn, initialPending } = this.props;
    if (initialPending) return <Spinner />;
    return (
      <ConnectedRouter history={history}>
        <Layout style={{ minHeight: '100vh' }}>
          <Switch>
            <Route exact path="/login" component={Login} />
            <Route exact path="/registration" component={Registration} />
            <ProtectedRoute isUserLoggedIn={isUserLoggedIn} path="/" component={MainLayout} />
          </Switch>
        </Layout>
      </ConnectedRouter>
    );
  }
}

function mapStateToProps(state) {
```

```
  return {
    isUserLoggedIn: state.user.isUserLoggedIn,
    initialPending: state.user.initialPending,
  };
}

function mapDispatchToProps(dispatch) {
  return {
    actions: bindActionCreators(
      {
        checkIsUserLoggedIn,
        getCourseList,
        getRoadmap,
      },
      dispatch,
    ),
  };
}

export default connect(mapStateToProps, mapDispatchToProps)(App);
```

src/registerServiceWorker.js

```
const isLocalhost = Boolean(
  window.location.hostname === 'localhost' ||
    window.location.hostname === '[::1]' ||
    window.location.hostname.match(
      /^127(?:\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)){3}$/
    )
);

export default function register() {
  if (process.env.NODE_ENV === 'production' && 'serviceWorker' in navigator) {
    // The URL constructor is available in all browsers that support SW.
```

```
    const publicUrl = new URL(process.env.PUBLIC_URL, window.location);
    if (publicUrl.origin !== window.location.origin) {
      return;
    }

    window.addEventListener('load', () => {
      const swUrl = `${process.env.PUBLIC_URL}/service-worker.js`;

      if (isLocalhost) {
        checkValidServiceWorker(swUrl);
        navigator.serviceWorker.ready.then(() => {
          console.log(
            'This web app is being served cache-first by a service ' +
              'worker. To learn more, visit https://goo.gl/SC7cgQ'
          );
        });
      } else {
        registerValidSW(swUrl);
      }
    });
  }
}

function registerValidSW(swUrl) {
  navigator.serviceWorker
    .register(swUrl)
    .then(registration => {
      registration.onupdatefound = () => {
        const installingWorker = registration.installing;
        installingWorker.onstatechange = () => {
          if (installingWorker.state === 'installed') {
            if (navigator.serviceWorker.controller) {
              console.log('New content is available; please refresh.');
            } else {
```

```
            console.log('Content is cached for offline use.');
          }
        }
      };
    };
  })
  .catch(error => {
    console.error('Error during service worker registration:', error);
  });
}

function checkValidServiceWorker(swUrl) {
 fetch(swUrl)
  .then(response => {
    if (
      response.status === 404 ||
      response.headers.get('content-type').indexOf('javascript') === -1
    ) {
      navigator.serviceWorker.ready.then(registration => {
        registration.unregister().then(() => {
          window.location.reload();
        });
      });
    } else {
      registerValidSW(swUrl);
    }
  })
  .catch(() => {
    console.log(
      'No internet connection found. App is running in offline mode.'
    );
  });
}
```

```
export function unregister() {
  if ('serviceWorker' in navigator) {
    navigator.serviceWorker.ready.then(registration => {
      registration.unregister();
    });
  }
}
```

src/view/CoursePage/index.js

```
import { connect } from 'react-redux';
import { withRouter } from 'react-router-dom';
import { bindActionCreators } from 'redux';
import { logout } from 'redux/user/actions';
import { getCourseData } from 'redux/courses/actions';
import CoursePage from './CoursePage';

function mapStateToProps(state) {
  return {
    courses: state.courses.courseList,
    courseData: state.courses.courseData,
  };
}

function mapDispatchToProps(dispatch) {
  return {
    actions: bindActionCreators(
      {
        logout,
        getCourseData,
      },
      dispatch,
    ),
  };
```

```
}

export default withRouter(connect(mapStateToProps, mapDispatchToProps)(CoursePage));
```

src/view/CreateCoursePage/index.js

```
import { connect } from 'react-redux';
import { withRouter } from 'react-router-dom';
import { bindActionCreators } from 'redux';
import { getSensorList } from 'redux/sensor/actions';
import { showNotification } from 'redux/notification/actions';
import CreateCoursePage from './CreateCoursePage';

function mapStateToProps(state) {
  return {
    sensors: state.sensor.sensors,
    actionTypes: state.sensor.actionTypes,
    arduinos: state.sensor.arduinos,
    places: state.sensor.places,
  };
}

function mapDispatchToProps(dispatch) {
  return {
    actions: bindActionCreators(
      {
        getSensorList,
        showNotification,
      },
      dispatch,
    ),
  };
}
```

```
export default withRouter(connect(mapStateToProps, mapDispatchToProps)(CreateCoursePage));
```

src/view/Dashboard/index.js

```
import { connect } from 'react-redux';
import { withRouter } from 'react-router-dom';
import { bindActionCreators } from 'redux';
import { logout } from 'redux/user/actions';
import Dashboard from './Dashboard';

function mapStateToProps(state) {
  return {
    user: state.user,
  };
}

function mapDispatchToProps(dispatch) {
  return {
    actions: bindActionCreators(
      {
        logout,
      },
      dispatch,
    ),
  };
}

export default withRouter(connect(mapStateToProps, mapDispatchToProps)(Dashboard));
```

src/view/LessonPage/index.js

```
import { connect } from 'react-redux';
import { withRouter } from 'react-router-dom';
import { bindActionCreators } from 'redux';
```

```
import { logout, trackActivity, completeLesson } from 'redux/user/actions';
import { getCourseData, getLessonData } from 'redux/courses/actions';
import { showNotification } from 'redux/notification/actions';
import LessonPage from './LessonPage';

function mapStateToProps(state) {
  const activeSection = state.courses.courseData.sections
    .find(el => el.lessons.indexOf(state.courses.lessonData['_id'] !== -1)) || {};
  return {
    course: state.courses.courseData.course || {},
    lessonData: state.courses.lessonData,
    courseData: state.courses.courseData,
    activeSection,
    activeSectionIndex: state.courses.courseData.sections.map(el => el['_id'])
      .indexOf(activeSection['_id']),
    user: state.user,
  };
}

function mapDispatchToProps(dispatch) {
  return {
    actions: bindActionCreators(
      {
        logout,
        getCourseData,
        getLessonData,
        showNotification,
        trackActivity,
        completeLesson,
      },
      dispatch,
    ),
  };
}
```

```
export default withRouter(connect(mapStateToProps, mapDispatchToProps)(LessonPage));
```

src/view/Login/index.js

```
import { connect } from 'react-redux';
import { withRouter } from 'react-router-dom';
import { bindActionCreators } from 'redux';
import { login } from 'redux/user/actions';
import LoginPage from './Login';

function mapStateToProps(state) {
  return {
    isUserLoggedIn: state.user.isUserLoggedIn,
  };
}

function mapDispatchToProps(dispatch) {
  return {
    actions: bindActionCreators(
      {
        login,
      },
      dispatch,
    ),
  };
}

export default withRouter(connect(mapStateToProps, mapDispatchToProps)(LoginPage));
```

src/view/Registration/index.js

```
import { connect } from 'react-redux';
import { withRouter } from 'react-router-dom';
```

```
import { bindActionCreators } from 'redux';
import { registration } from 'redux/user/actions';
import RegistrationPage from './Registration';

function mapStateToProps(state) {
  return {
    isUserLoggedIn: state.user.isUserLoggedIn,
  };
}

function mapDispatchToProps(dispatch) {
  return {
    actions: bindActionCreators(
      {
        registration,
      },
      dispatch,
    ),
  };
}

export default withRouter(connect(mapStateToProps, mapDispatchToProps)(RegistrationPage));
```

src/view/Roadmap/index.js

```
import { connect } from 'react-redux';
import { withRouter } from 'react-router-dom';
import { bindActionCreators } from 'redux';
import { logout } from 'redux/user/actions';
import Roadmap from './Roadmap';

function mapStateToProps(state) {
  return {
    roadmap: state.courses.roadmap,
```

```
  };
}

function mapDispatchToProps(dispatch) {
  return {
    actions: bindActionCreators(
      {
        logout,
      },
      dispatch,
    ),
  };
}

export default withRouter(connect(mapStateToProps, mapDispatchToProps)(Roadmap));
```

src/view/UserPage/index.js

```
import { connect } from 'react-redux';
import { withRouter } from 'react-router-dom';
import { bindActionCreators } from 'redux';
import { logout, updateUser } from 'redux/user/actions';
import UserPage from './UserPage';

function mapStateToProps(state) {
  return {
    user: state.user,
  };
}

function mapDispatchToProps(dispatch) {
  return {
    actions: bindActionCreators(
      {
```

```
        logout, updateUser,
      },
      dispatch,
    ),
  };
}

export default withRouter(connect(mapStateToProps, mapDispatchToProps)(UserPage));
```

ДОДАТОК В

Освітня web-система моделювання процесів створення розумного будинку з елементами віртуальної реальності

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС ТВз5104_18Б _

Аркушів 10

Київ 2018

АНОТАЦІЯ

Додаток містить опис освітньої web-система моделювання процесів створення розумного будинку з елементами віртуальної реальності, що виконує деякі із завдань, поставлених в розділі 1, а саме:

— Побудова інтерфейсу для створення власних курсів

— Реалізація зручного інтерфейсу для доступу до функціоналу додатку.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку міститься опис основного компоненту освітньої web-система моделювання процесів створення розумного будинку з елементами віртуальної реальності, що виконує деякі із завдань, поставлених в розділі 1. У додатку Б міститься програмний код компоненту.

Система для роботи потребує браузер, що підтримує JavaScript та Cookies.

Додаток розроблений за допомогою мови програмування Javascript.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Додаток надає функціонал для створення власних анімацій сцен віртуальної реальності, а саме:

— вибір версії Arduino;

— вибрати типи сенсорів;

— вибрати кількість сенсорів;

— поєднати обрані сенсори між собою;

— обрати дії, які має виконати актор;

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Згідно з архітектурою, додаток складається з трьох головних модулів:

— Модуль відображення списку курсів;

— Модуль відображення уроку;

— Модуль створення анімації.

Модуль відображення списку курсів містить в собі класи для обробки даних про курси і прогрес користувача відносно них.

Модуль відображення уроку містить в собі класи для відображення даних про урок, обраний етап, .прогрес по курсу.

Модуль створення анімації містить в собі класи для введення і обробки даних від користувача для створення нової сцени віртуальної реальності.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для використання додатку користувач повинен мати персональний комп’ютер або мобільний пристрій з браузером, що підтримує JavaScript, а також підключення до мережі Інтернет.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Програма не потребує інсталяції і доступна на сайті за посиланням http://localhost:8080.

ВХІДНІ ДАНІ

Вхідна інформація для додатку:

а) дані для створення індивідуального облікового запису;

ВИХІДНІ ДАНІ

Вихідна інформація додатку:

а) курси, запропоновані користувачу

б) конструктор анімацій.